Resumo Grupo 3

Bernardo Ribeiro Effgen
Bryan Zanelato
Cauã Agrisi Merisio
Gabriel Mendonça Leão Borges
João Victor Igreja Mozer

23 de abril de 2026

Resumo

Resumo do capítulo 4.4 (páginas 110-120) do livro Game Engine Black Book: Wolfenstein 3D, de Fabien Sanglard.

Sumário

1 Capítulo 4.4 - Architecture

Esse capítulo introduz a estrutura geral do motor do Wolfenstein 3D, mostrando que ele foi projetado de forma organizada e modular. O código é dividido em duas camadas principais. A primeira é a camada WL_, que representa o nível mais alto do sistema, onde ficam as regras do jogo, como movimentação do jogador, lógica dos inimigos, menus e fluxo geral. A segunda camada é formada pelos Managers (ID_), que são módulos responsáveis por lidar com tarefas específicas e mais próximas do hardware, como memória, vídeo e som. Essa separação é extremamente importante porque permite que o código do jogo não precise se preocupar diretamente com detalhes técnicos do hardware, tornando o sistema mais organizado e reutilizável.

1.1 Gerenciamento de Memória e Engenharia de Software

O sistema de gerenciamento de memória (*Memory Manager* - MM) do *Wolfenstein 3D* é um exemplo notável de engenharia pragmática para contornar as limitações do ambiente DOS e os 640 KB de memória convencional. Diferente do `malloc` padrão da biblioteca C, que é suscetível à fragmentação por não permitir o deslocamento de blocos após a alocação, o motor da *id Software* implementa um sistema customizado baseado em uma lista duplamente encadeada (*Linked List*).

1.1.1 Estrutura e Inicialização

O motor reserva quase a totalidade da RAM disponível para criar seu próprio *heap*. O controle é feito por dois ponteiros principais: o **Head**, que marca o início da lista, e o **Rover**, um ponteiro móvel que otimiza a busca por espaço livre, evitando a iteração completa da lista em cada requisição.

1.1.2 Processo de Alocação em Três Fases

A alocação de memória segue níveis crescentes de complexidade técnica:

1. **Busca Direta:** Verifica-se a existência de espaço livre imediatamente após o ponteiro *Rover*. É a operação mais rápida do sistema.

2. **Busca e Purga:** O *Rover* percorre a lista identificando blocos marcados como removíveis (recursos como áudio ou texturas que podem ser recarregados do disco). Estes são liberados até que se obtenha um bloco contínuo suficiente.

3. **Compactação (Desfragmentação):** Em última instância, o motor move todos os blocos ocupados em direção ao início da memória, eliminando os espaços vazios intermitentes.

1.1.3 Limitações e Bloqueios (*Locking*)

Um aspecto crítico é o tratamento de blocos marcados como `LOCKED`. Se o algoritmo de compactação encontra um bloco travado, ele interrompe o deslocamento e reinicia o processo apenas após esse bloco. Embora essa decisão de *design* possa resultar em desperdício de espaço, ela simplifica drasticamente a lógica da *engine*, evitando a complexidade de gerenciar referências de memória móveis em torno de dados estáticos.

1.2 Page Manager (PM): Paginação e Cache de Ativos

O *Page Manager* é o componente responsável pela orquestração do fluxo de dados entre o armazenamento secundário (HDD) e a memória RAM. Inspirado em conceitos de memória virtual de sistemas Unix, o PM trata recursos como texturas, *sprites* e sons como páginas fixas de 4 KiB, utilizando *Asset IDs* gerados pela ferramenta `IGRAB` para abstrair os endereços físicos.

1.2.1 Hierarquia de Cache e Memória

Para otimizar a performance sob as restrições do hardware da época, o sistema implementa uma arquitetura de cache em dois níveis:

- **Cache L1:** Composta pela memória Convencional e EMS (*Expanded Memory Specification*), representando o nível de acesso mais veloz.
- **Cache L2:** Utiliza a memória XMS (*Extended Memory Specification*) como um reservatório intermediário antes de recorrer ao disco.

1.2.2 Ciclo de Vida do Ativo e Thrashing

O processo de carregamento inicial, visualmente identificado pela tela *"Get Psyched!"*, realiza o *precaching* linear dos dados contidos no arquivo `VSWAP.WL1`. Durante a execução, o PM aplica a política **LRU** (*Least Recently Used*) para a substituição de páginas.

Um desafio crítico deste projeto é o fenômeno de *Thrashing*, que ocorre quando a demanda por recursos em um único quadro excede a capacidade da memória, forçando o sistema a carregar e descartar o mesmo ativo repetidamente, resultando em quedas severas de *framerate*.

1.2.3 Exceção Crítica: Áudio

Diferente dos ativos gráficos, os dados de áudio são tratados como exceções operacionais. Devido à dependência de interrupções de hardware para a reprodução, os sons são carregados prioritariamente e mantidos exclusivamente na memória convencional, evitando a latência do sistema de *swapping* que causaria falhas ou atrasos na saída sonora.

1.3 Video Manager: Abstração de Hardware e Renderização

O gerenciamento de vídeo em *Wolfenstein 3D* é estruturado em uma arquitetura de duas camadas, projetada para equilibrar a necessidade de alta performance com a produtividade no desenvolvimento. Essa separação permite que operações críticas de desenho sejam otimizadas sem expor a complexidade do hardware para toda a lógica do jogo.

1.3.1 Arquitetura de Camadas

- **VL (Video Low-level):** Representa a camada de base, comunicando-se diretamente com o hardware através de uma combinação de linguagens C e *Assembly*. É responsável por manipular registros da placa VGA e realizar operações de escrita direta na memória de vídeo.

- **VH (Video High-level):** Funciona como uma interface de abstração que consome os serviços da VL. Ela fornece aos programadores funções simplificadas para o desenho de elementos na tela (como menus, indicadores do HUD e textos), eliminando a necessidade de gerenciar detalhes técnicos de baixo nível durante o desenvolvimento de alto nível.

1.3.2 Benefícios do Design

Essa divisão modular foi fundamental para a portabilidade e manutenção do motor. Enquanto a **VL** garante que o motor extraia o máximo de desempenho do hardware disponível, a **VH** permite que a equipe de criação foque na interface e na experiência do usuário, tratando a tela como um plano abstrato de desenho em vez de um conjunto complexo de registros de hardware.

1.4 User Manager (US): Gerenciamento de Layout de Texto

Desenvolvido por Jason Blochowiak, o *User Manager* do *Wolfenstein 3D* é um exemplo de reaproveitamento de código legatário, herdando grande parte de sua estrutura do título anterior da *id Software*, o *Catacomb 3-D*.

1.4.1 Funcionalidade e Abstração

Apesar da nomenclatura sugerir o controle de entradas ou perfis de usuário, a função primária deste componente é o **gerenciamento de layout de texto**. Ele atua como uma camada lógica que precede a renderização visual:

- O motor solicita a exibição de uma *string*.
- O *User Manager* calcula as métricas de posicionamento, como o alinhamento e a centralização do texto na tela.
- Essas coordenadas e metadados são repassados ao *Video Manager*, que executa a renderização dos glifos.

1.4.2 Desenvolvimento e Código Legatário

Um aspecto curioso deste módulo é a evidência de um ciclo de desenvolvimento acelerado. O arquivo de cabeçalho apresenta diversas declarações de funções que nunca chegaram a ser implementadas na versão final do jogo, servindo como um registro histórico do pragmatismo da equipe, que optou por manter apenas o essencial para a funcionalidade de exibição de texto e interface básica.

1.5 Sound Manager: Processamento via Interrupção

O *Sound Manager* atua como uma camada de abstração crítica entre a lógica do jogo e os diversos dispositivos de áudio da época (como *PC Speaker*, *AdLib* e *Sound Blaster*). Sua arquitetura é projetada para garantir a fidelidade sonora independente das oscilações na taxa de quadros do motor principal.

1.5.1 Desacoplamento e Frequência de Amostragem

Diferente de outros módulos, o processamento de áudio não é executado de forma síncrona dentro do *loop* principal do jogo. Enquanto a renderização visual e a lógica ocorrem a aproximadamente 70 Hz, o *Sound Manager* opera em frequências muito superiores, podendo atingir até 7000 Hz.

1.5.2 Mecanismo de IRQ (*Interrupt Request*)

Para alcançar essa alta taxa de atualização sem sobrecarregar a CPU, o sistema utiliza interrupções de hardware (**IRQ**). Este mecanismo permite que:

- O computador envie sinais prioritários que interrompem momentaneamente o processamento atual para tratar dados de áudio.
- A reprodução de sons e músicas permaneça fluida, evitando artefatos sonoros ou "cortes"(*stuttering*), mesmo em situações onde a performance do motor gráfico sofre quedas.

1.5.3 Otimização e Prioridade

Dada a natureza sensível ao tempo do sinal sonoro, o gerenciador é implementado com foco em baixa latência. Essa prioridade via hardware garante que a experiência auditiva do usuário seja contínua, uma solução técnica essencial para a imersão em um ambiente de jogo acelerado como o de *Wolfenstein 3D*.

1.6 Input Manager (IN): Abstração de Periféricos

O *Input Manager* é o módulo responsável por mediar a comunicação entre o motor de jogo e os diversos dispositivos de entrada suportados, garantindo que a lógica de movimentação e combate seja independente do hardware específico utilizado pelo jogador.

1.6.1 Abstração de Hardware e Portas de I/O

Para oferecer suporte multiplataforma (dentro do ecossistema de PCs da época), o gerenciador lida com três padrões distintos de conexão, cada um operando em endereços de entrada e saída (*I/O addresses*) específicos:

- **Teclado (Porta PS/2):** O dispositivo primário de controle.
- **Mouse (Porta Serial):** Utilizado para navegação em menus e, opcionalmente, para o movimento de rotação no jogo.
- **Joystick (Porta DA-15):** Suporte para controles analógicos através da *Game Port*.

1.6.2 Camada Intermediária

A principal função deste componente é a padronização. Ele serve como uma camada técnica intermediária que traduz sinais elétricos e interrupções de hardware brutos em comandos lógicos que o motor consegue processar. Essa abstração permite que o código principal do jogo verifique estados como "Mover para Frente"sem precisar conhecer se o sinal veio de uma tecla pressionada ou do eixo de um *joystick*.